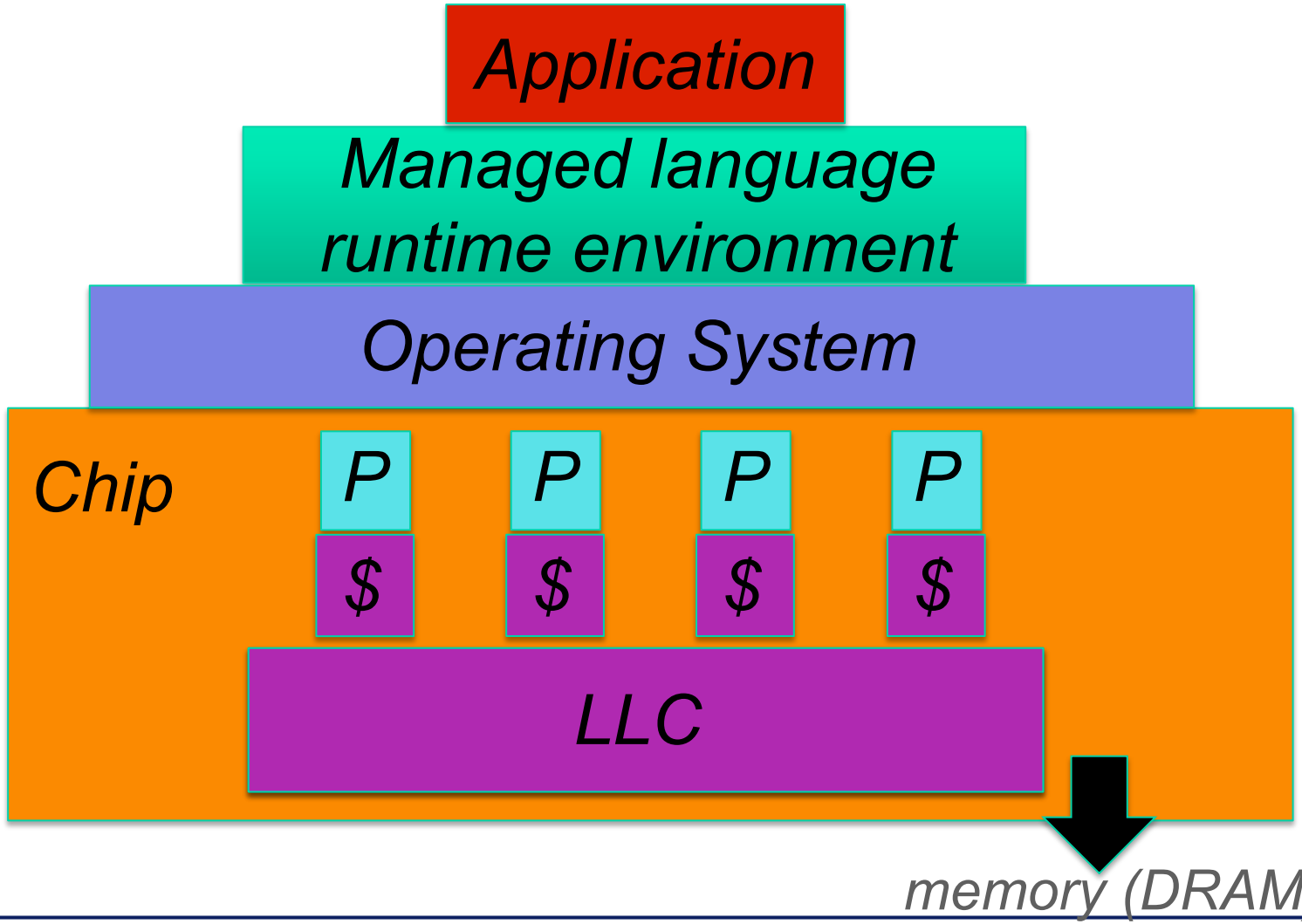

# Harnessing Memory Management to Optimize for Efficiency

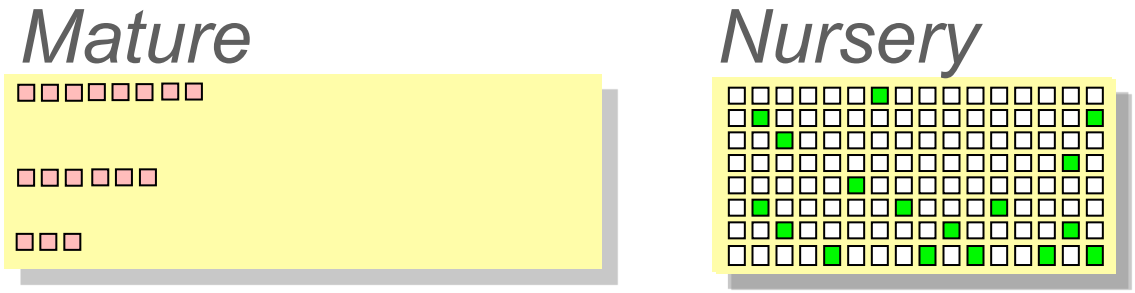
**Jennifer B. Sartor**

**Virtual Machine Summer School 2016**

# Multicore Challenge

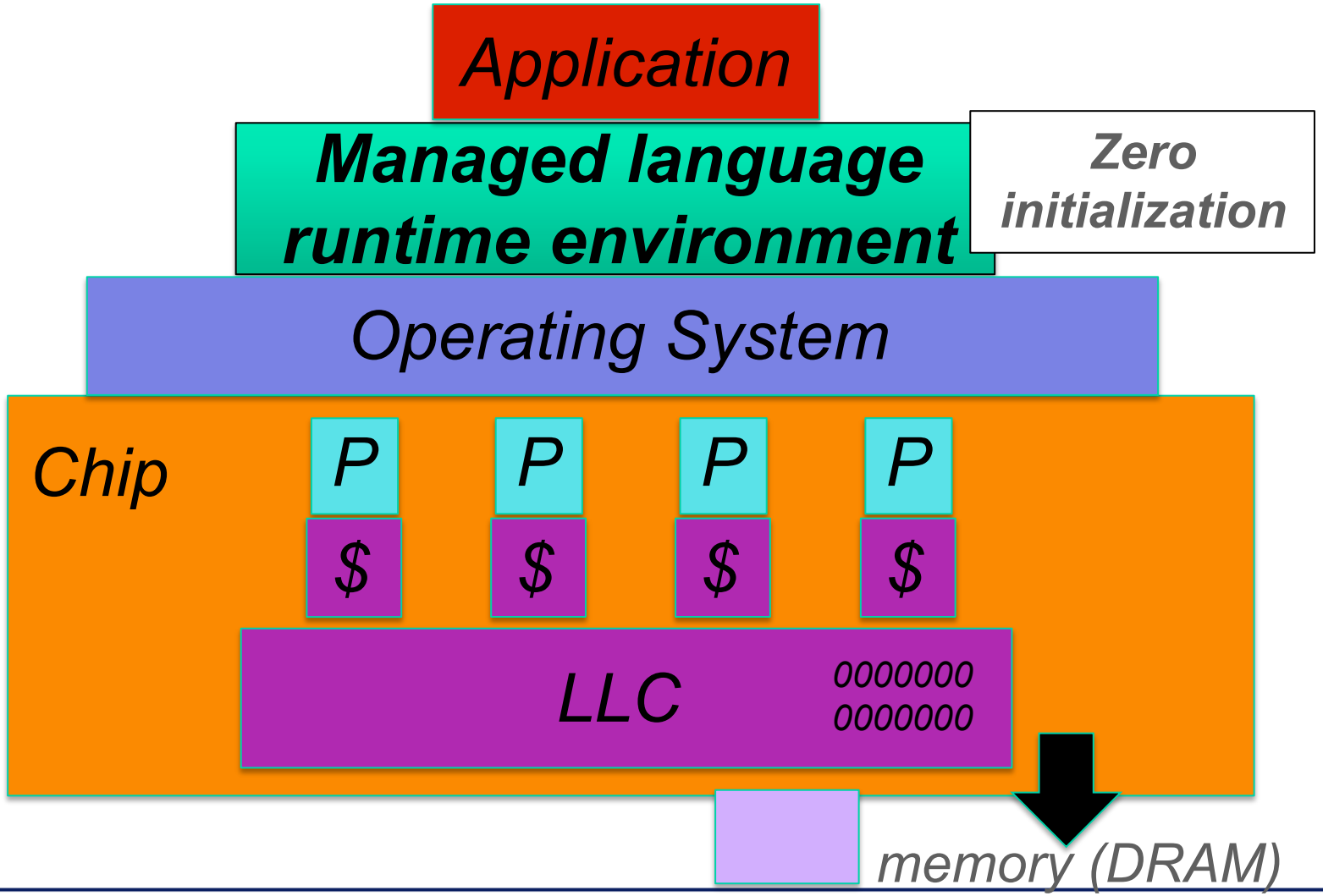


# Generational Garbage Collection

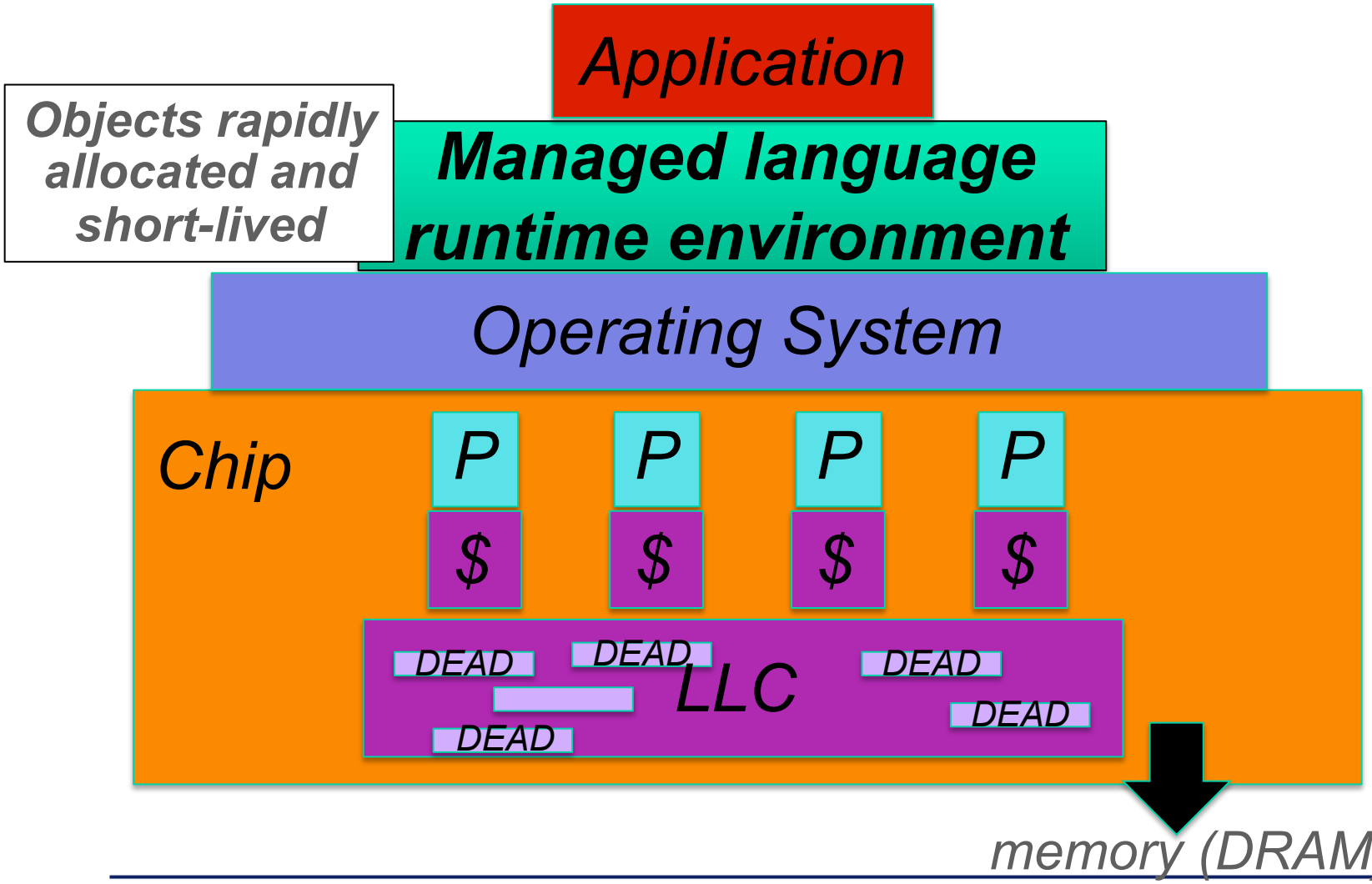



- **Young objects die quickly**
- **Nursery**
  - Traced for live objects
  - Copy to mature space
  - Reclaimed ‘en masse’

# Problem: Bandwidth & Power Wall



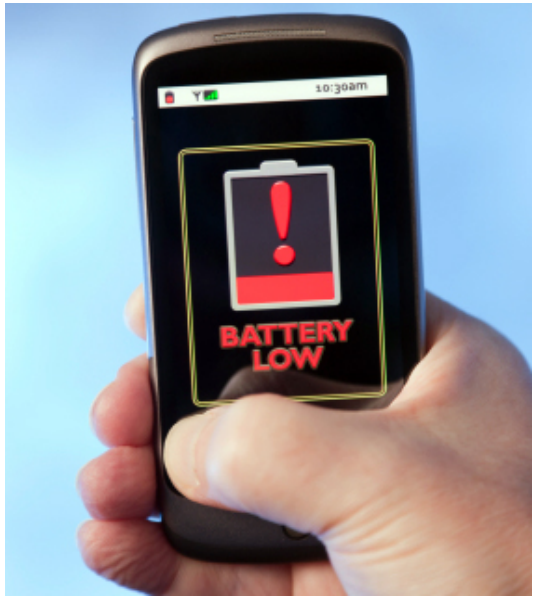
# Problem: Allocation Wall



# Why energy-efficient computing?

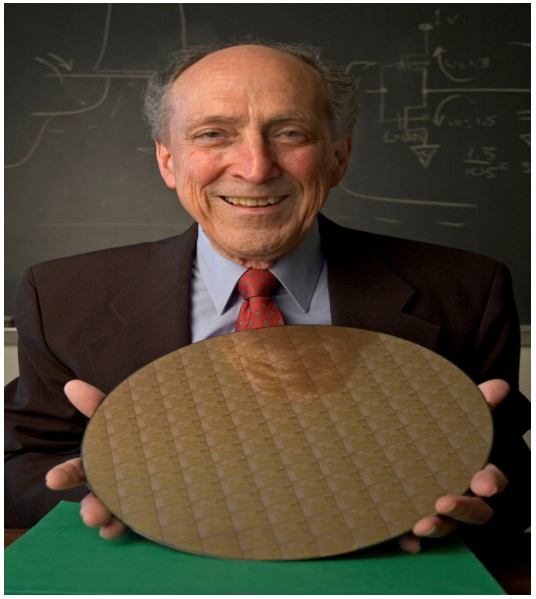
(1) 100 Billion kW.h per year in U.S. alone
(2) Saving 20% in efficiency = $2 billion

(energy.gov)




(1) More searches on mobile
(2) Battery life is a big concern

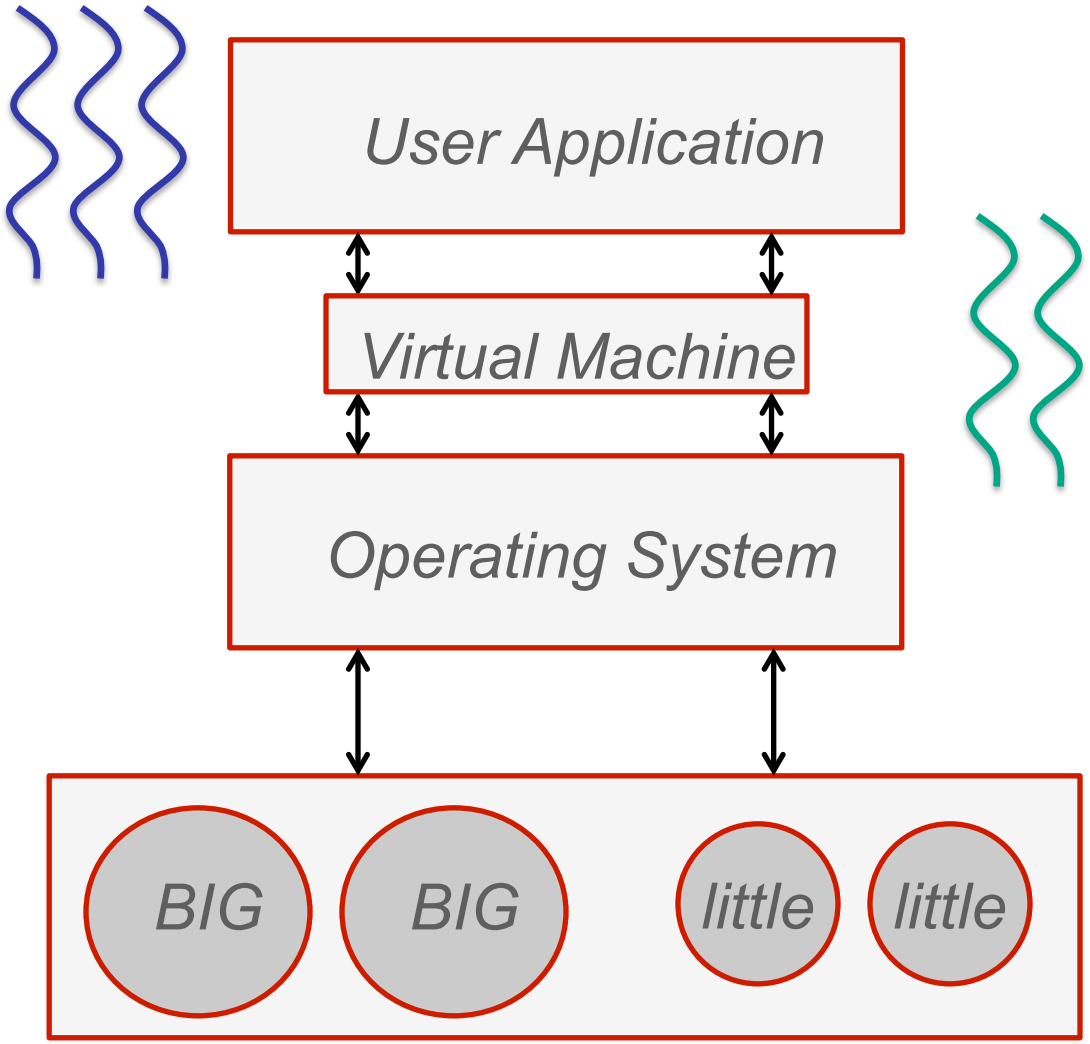
(Google)



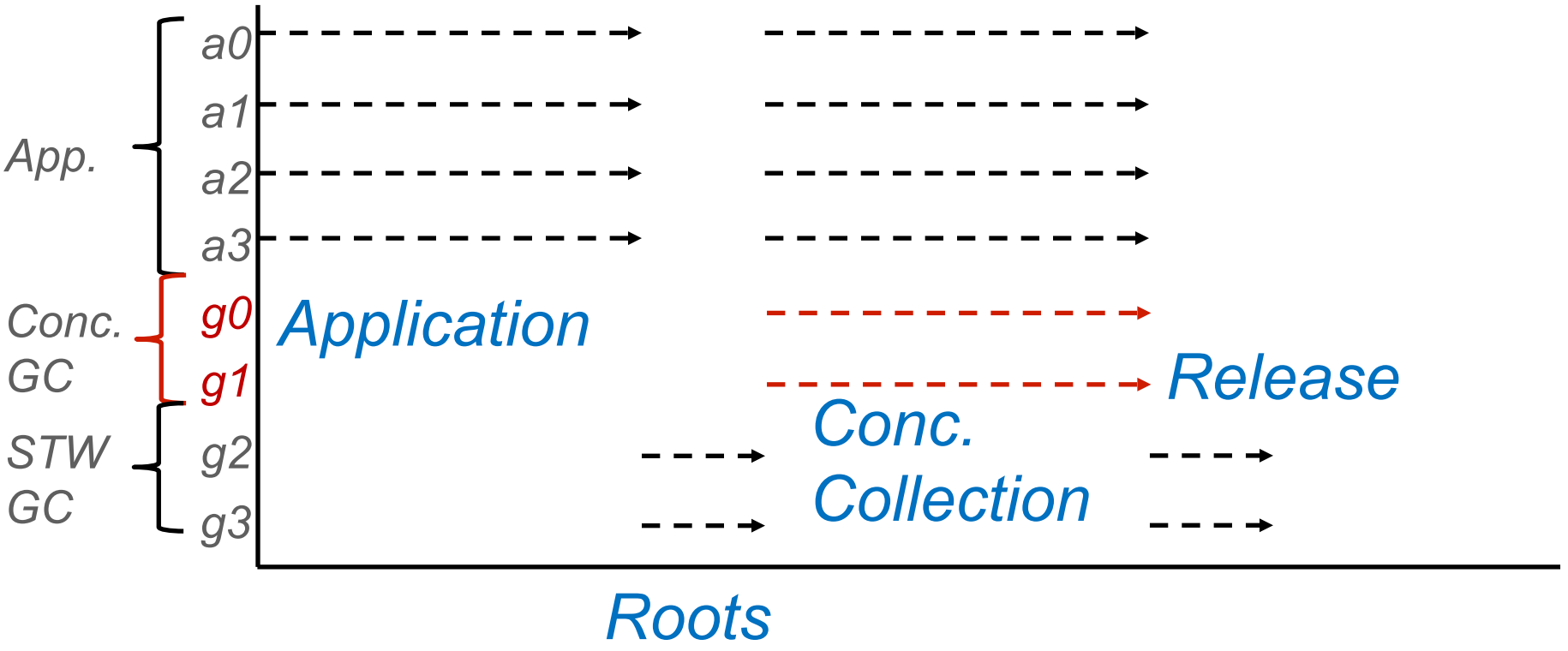
End of Dennard scaling makes modern chips power-constrained

(Robert H. Dennard)

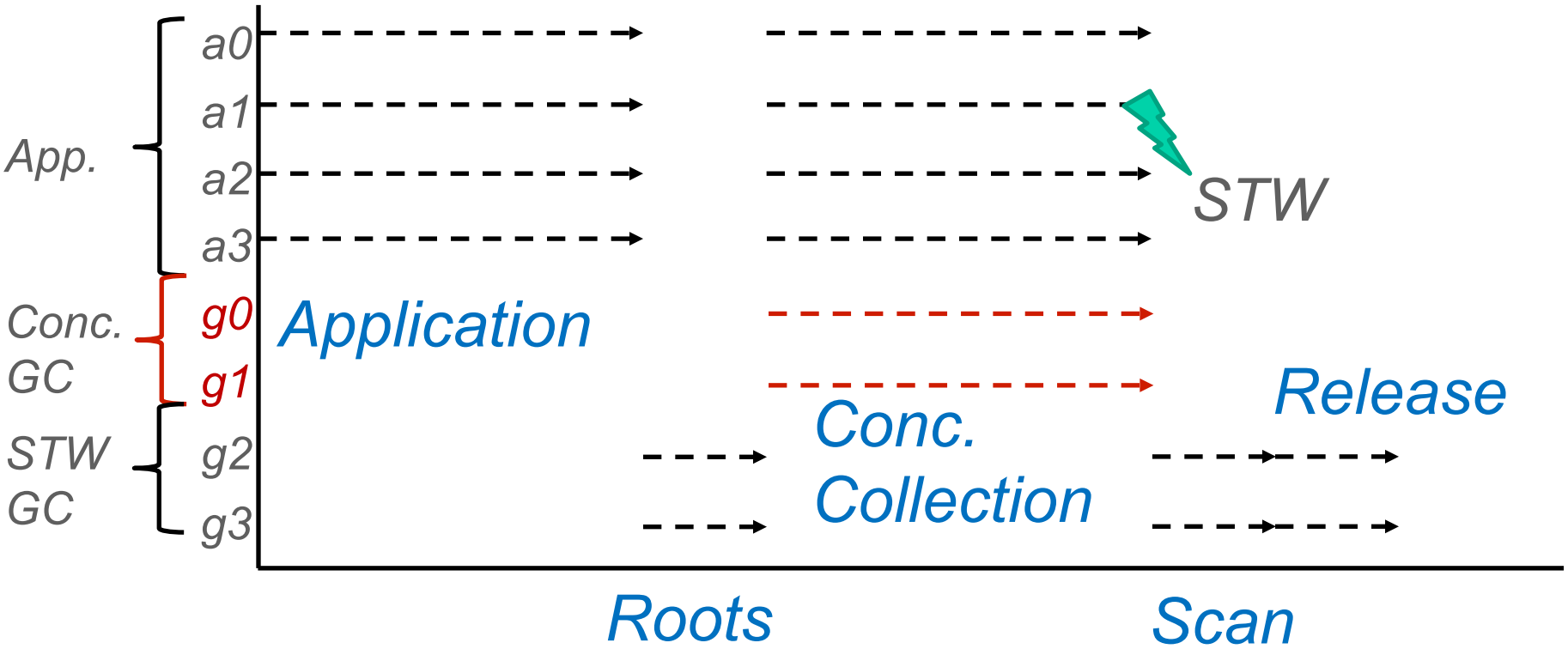
# Heterogeneous Multicores



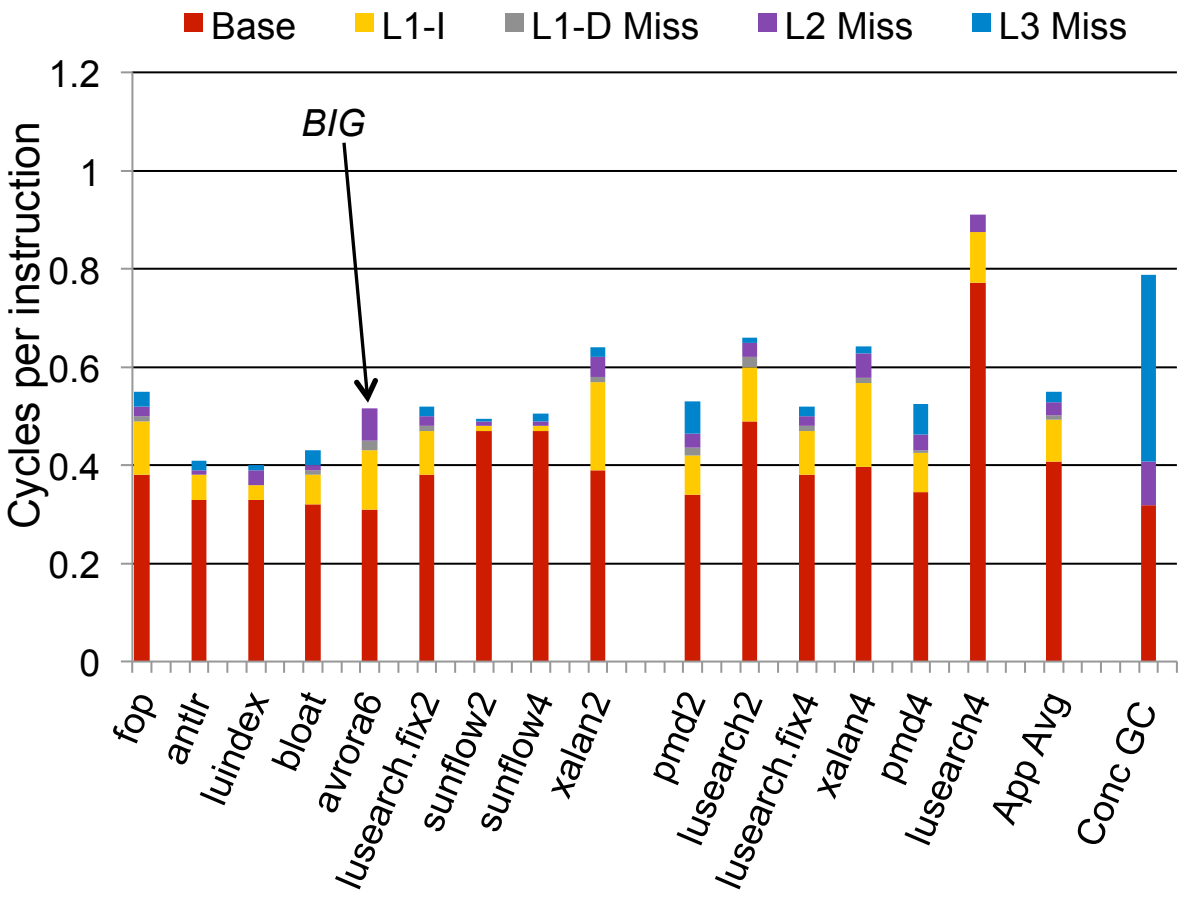
# Concurrent Garbage Collector



# If Collector Cannot Keep Up

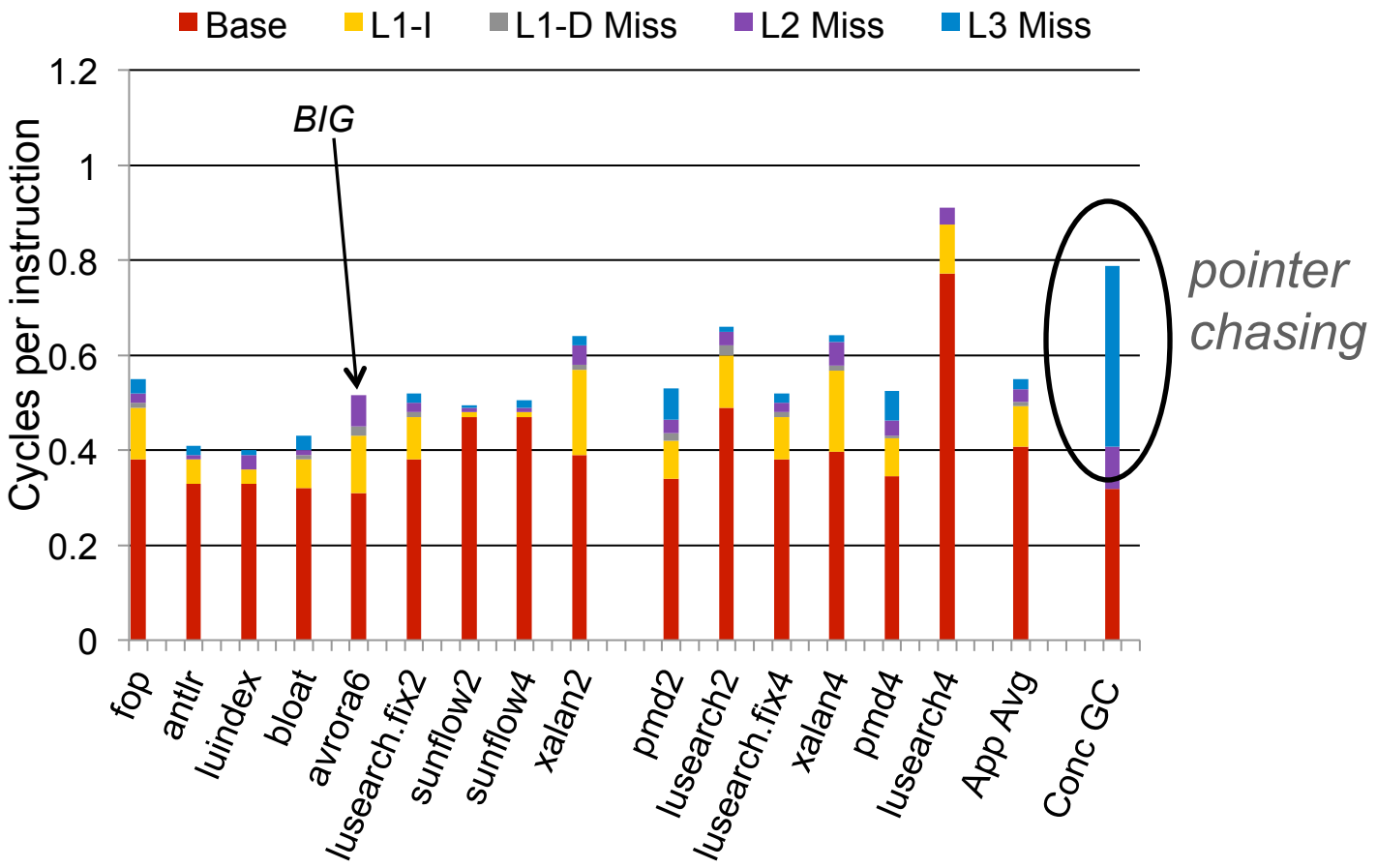


# CPI Stacks



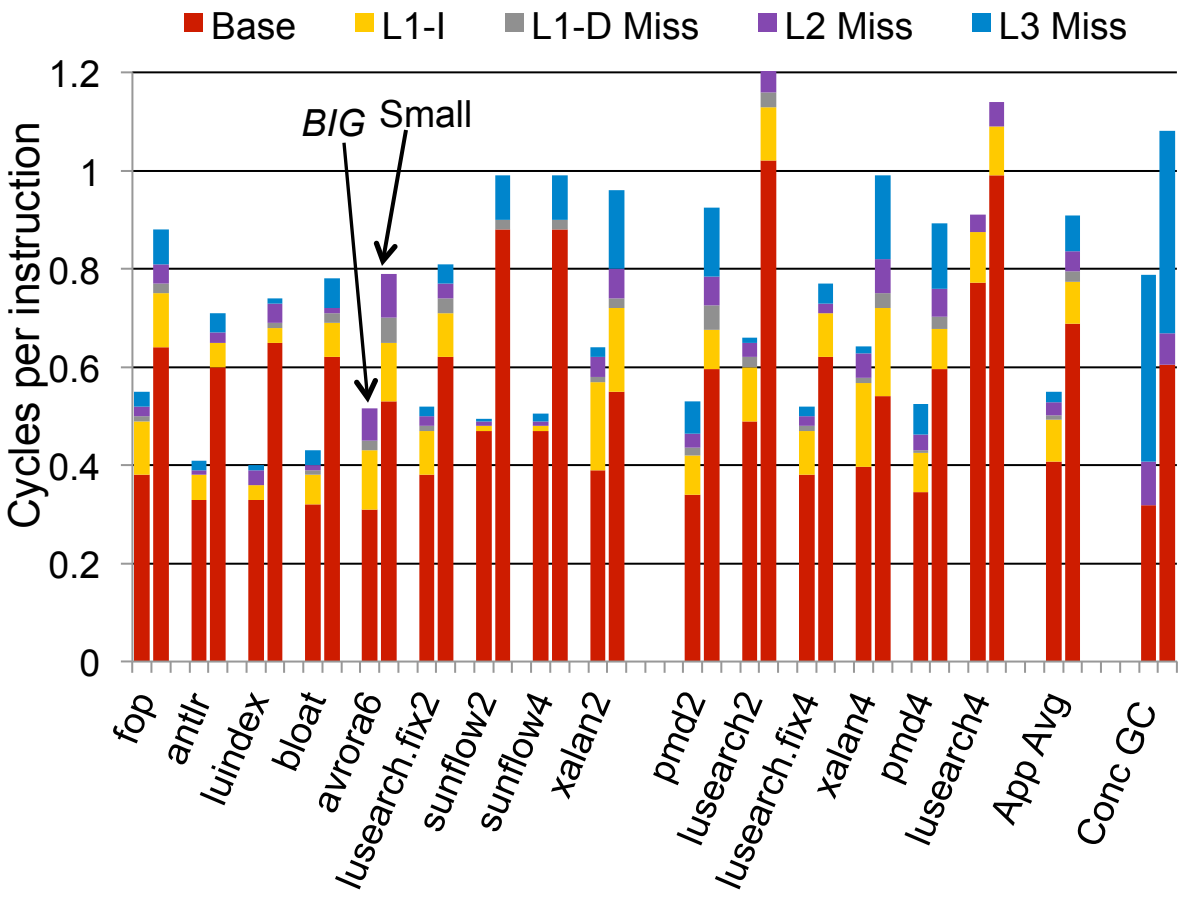

*Understanding the BIG core's performance advantage*

# CPI Stacks



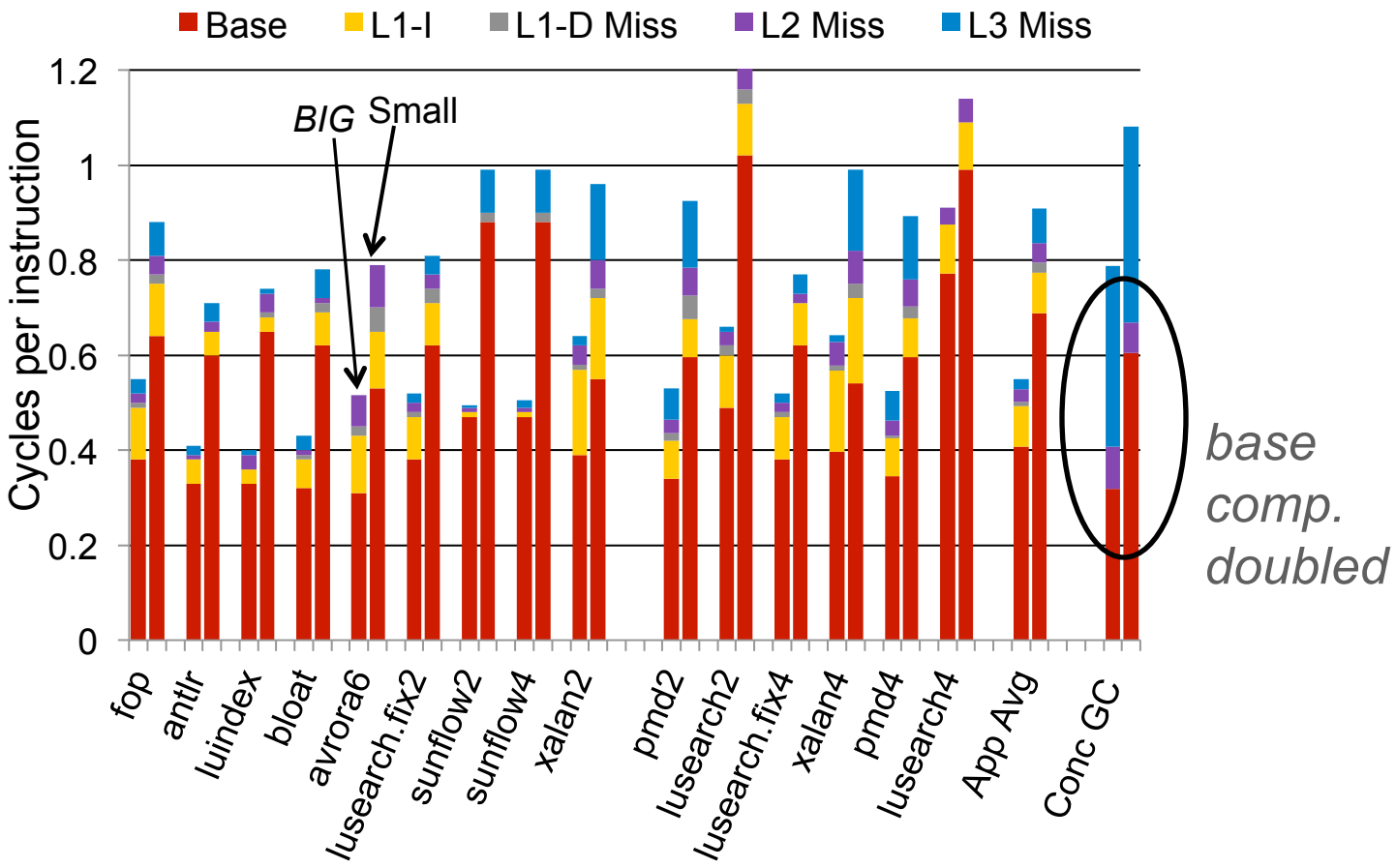

*Understanding the BIG core's performance advantage*

# CPI Stacks



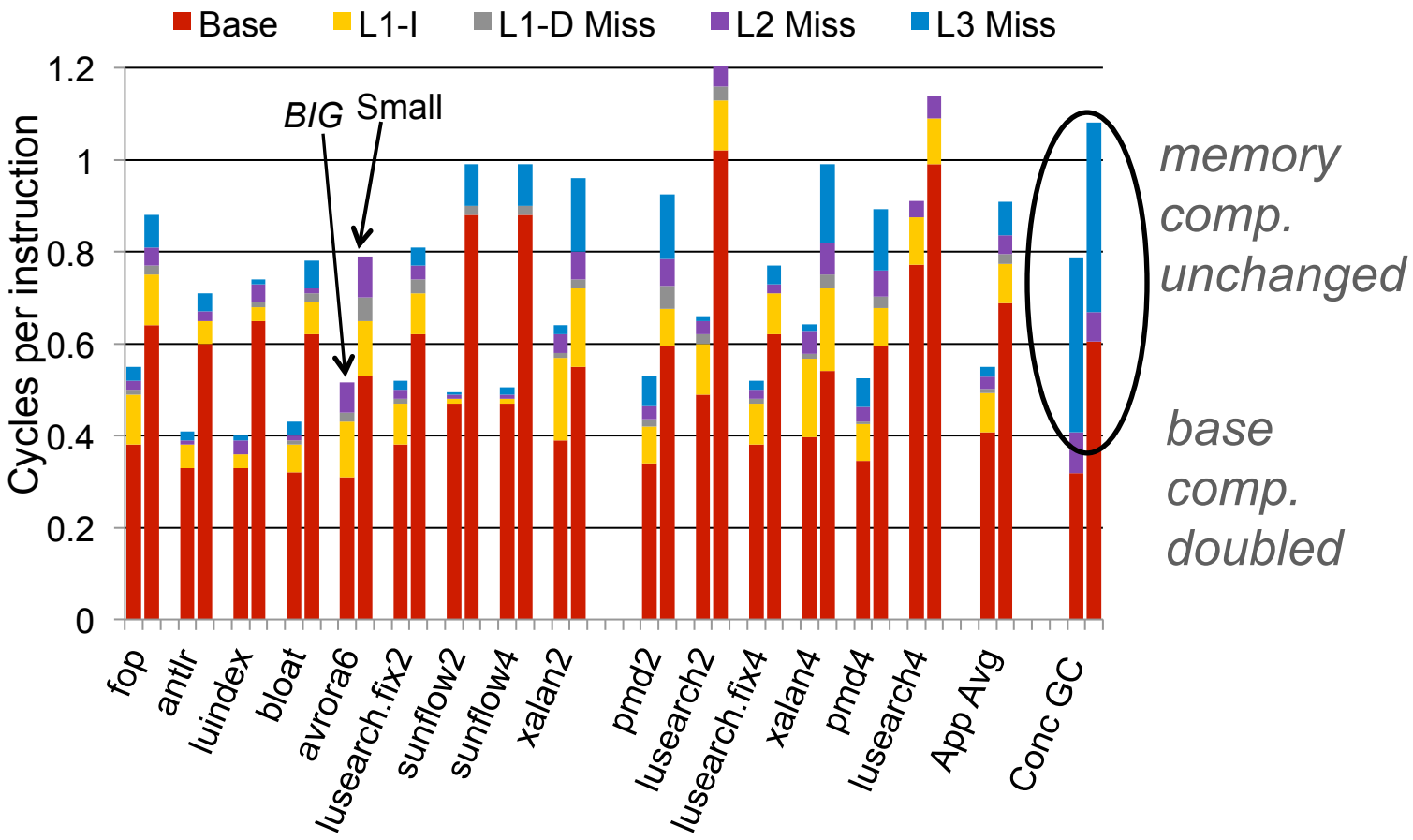

*Understanding the BIG core's performance advantage*

# CPI Stacks



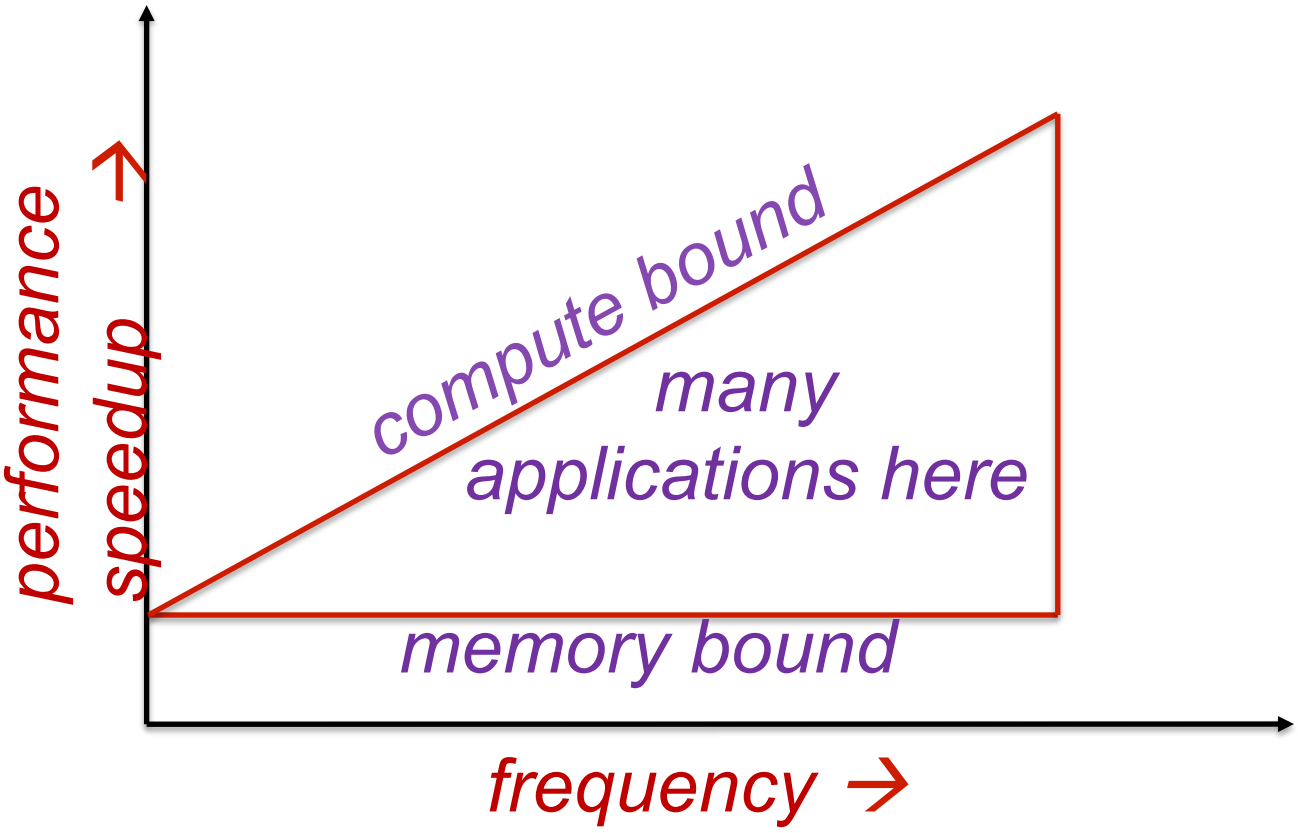

*Understanding the BIG core's performance advantage*

# CPI Stacks



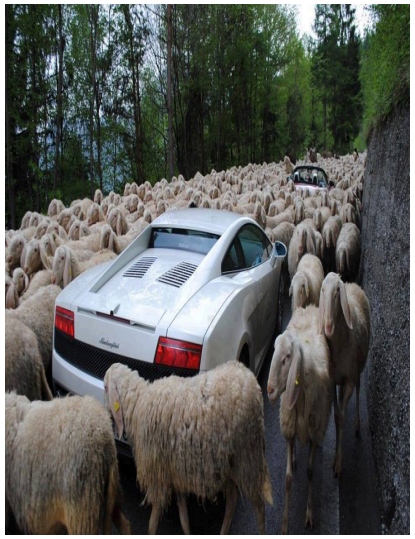

*Understanding the BIG core's performance advantage*

# DVFS Performance Prediction




Sample at all DVFS states ☹
**Estimate** performance ☺

# Managed Multi-threaded Applications

Heterogeneity

Synchronization

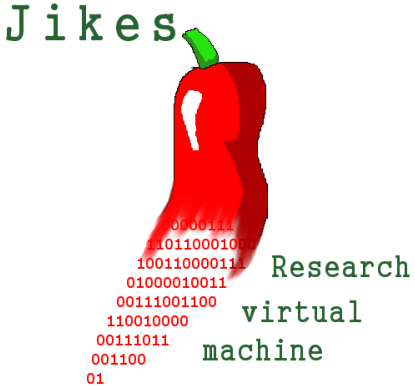
Store Bursts

# Sniper Simulator

- **Extensions to work with JVM**
  - Works with JIT compiler
  - Emulate system calls (futex & nanosleep)
  - JVM-simulator communication with new instruction
- Simulates
  - x86, cycle-level, parallel, high-speed
  - Multicore, heterogeneous
  - Different frequencies
  - McPat for power

# Methodology

- **Sniper simulator**
- **Jikes RVM 3.1.2 and DaCapo benchmarks**
  - Collector
    - **Generational Immix garbage collector**
    - **Concurrent mark-sweep snapshot algorithm**
  - 2x minimum heap
  - Replay compilation, 2nd invocation





# Cooperative Cache Scrubbing

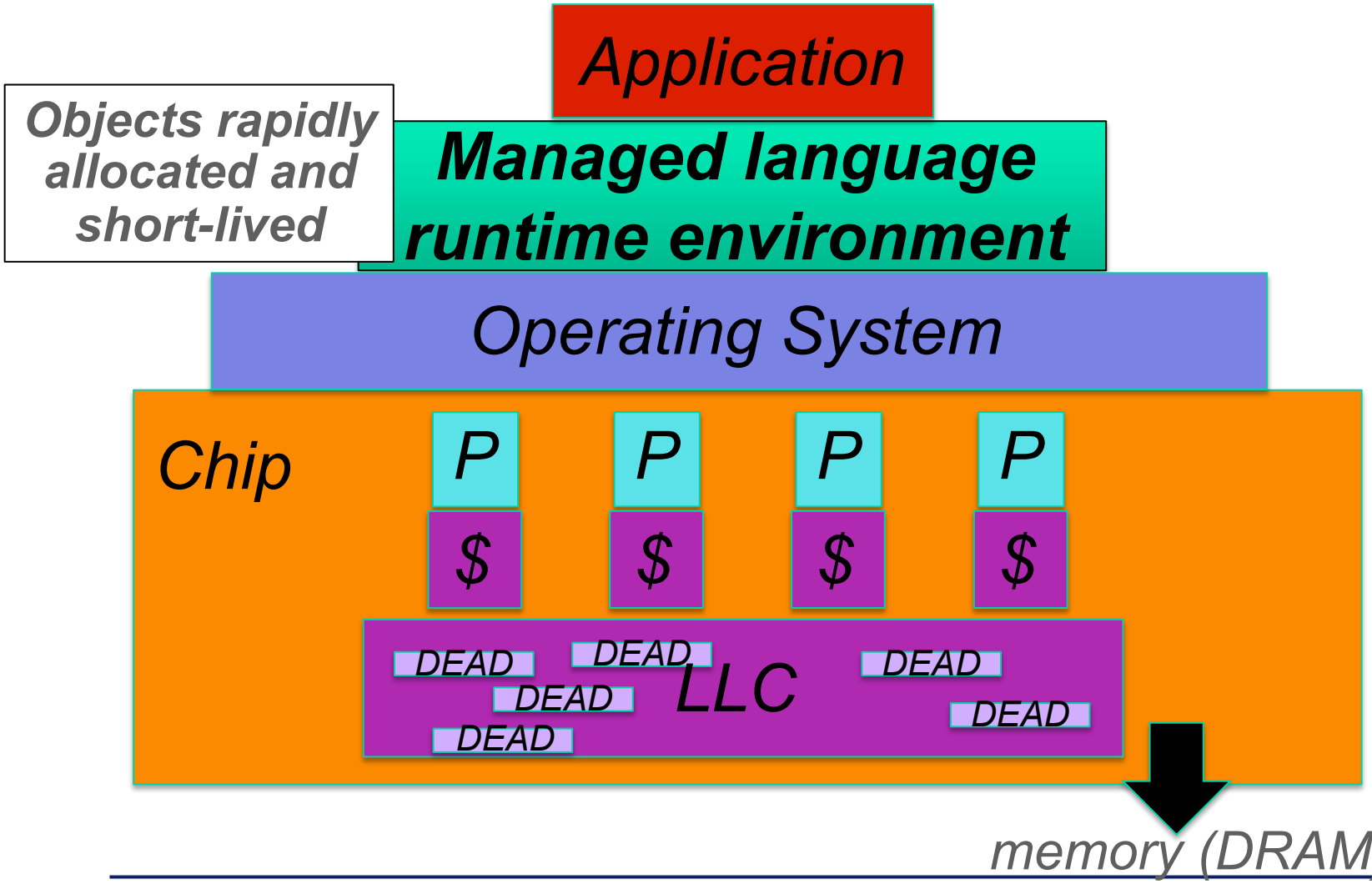
**Jennifer B. Sartor,** Wim Heirman, Steve Blackburn*, Lieven Eeckhout, Kathryn S. McKinley^

*PACT 2014*

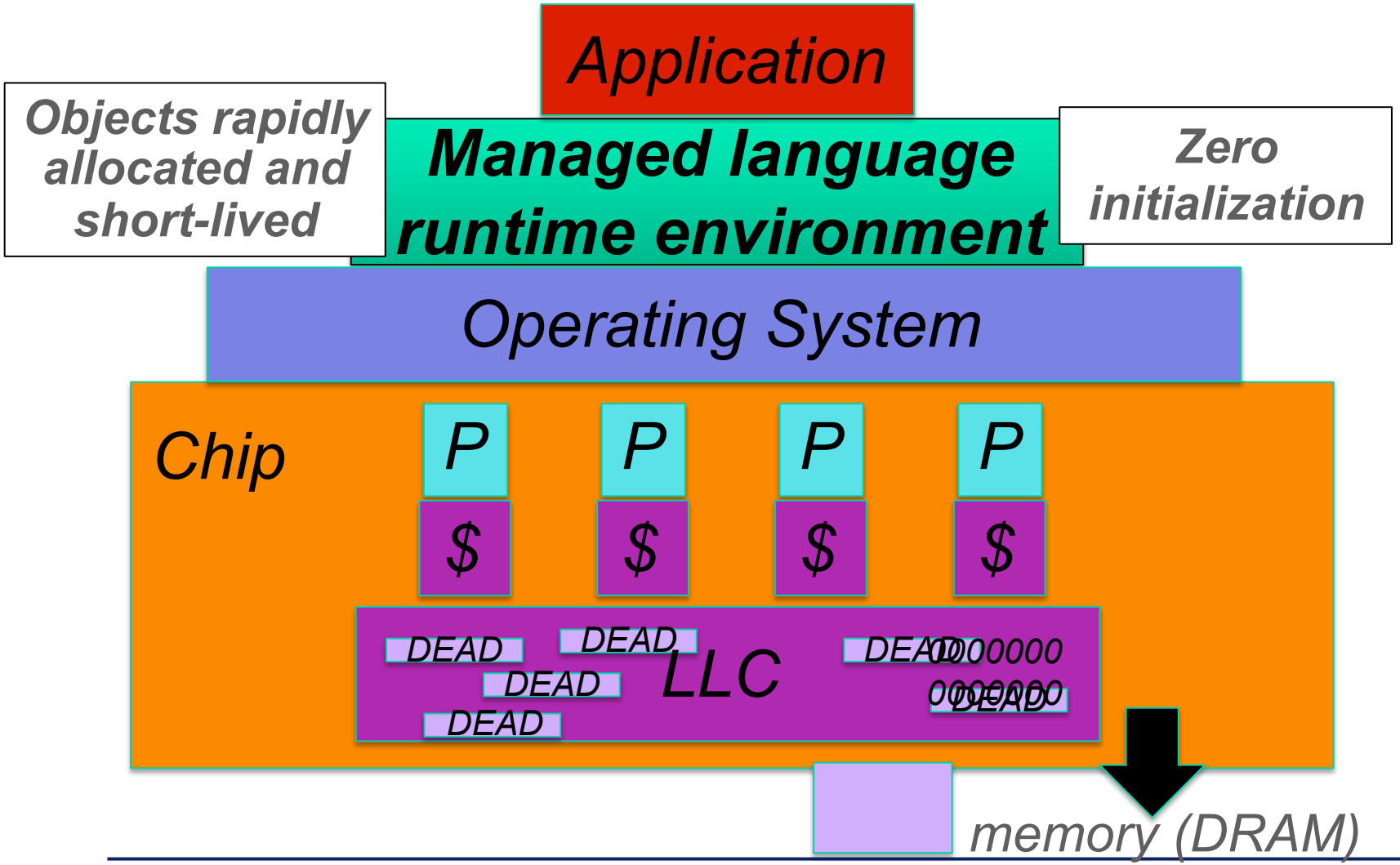
* Australian National University

^ Microsoft

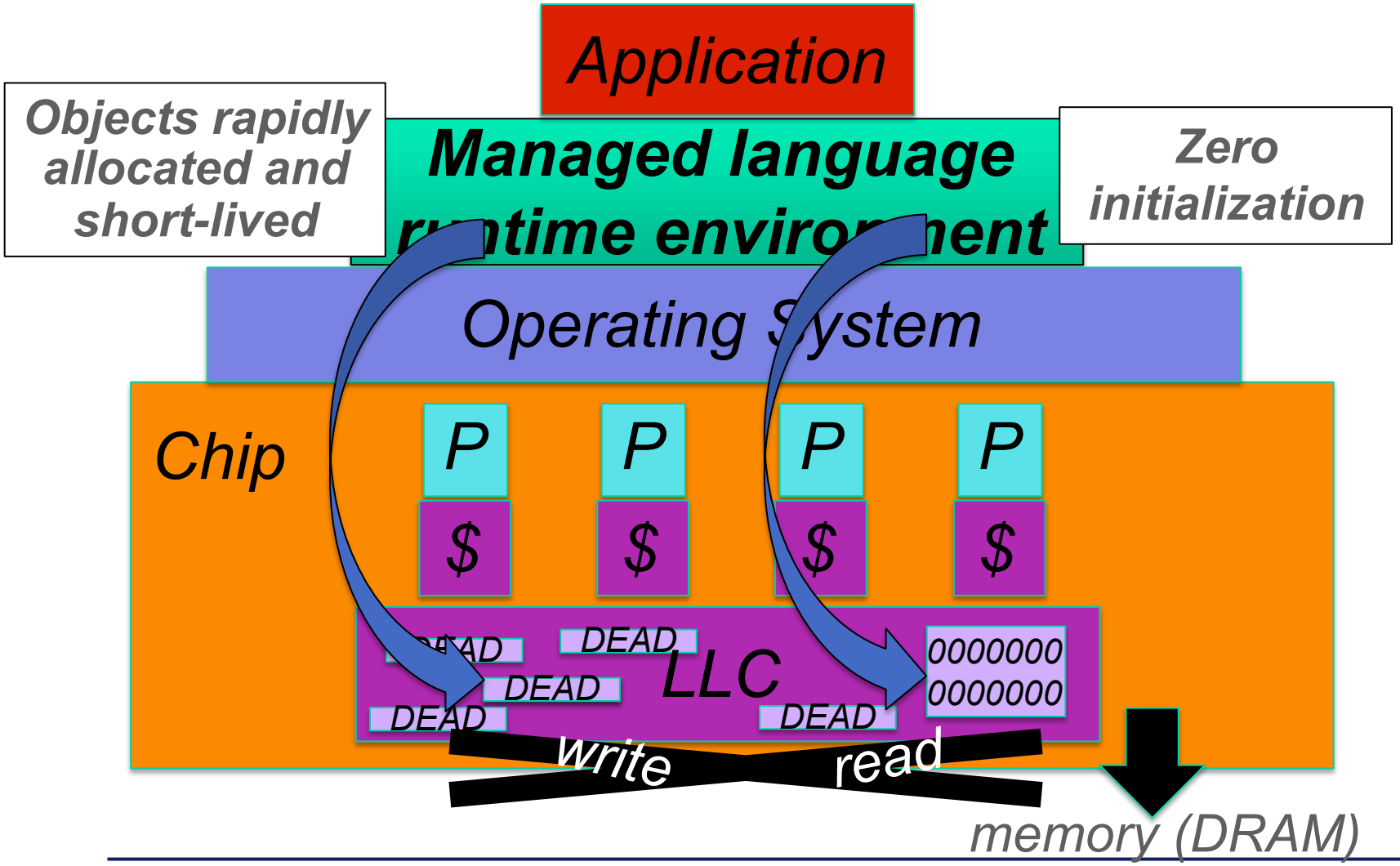
# Problem: Allocation Wall



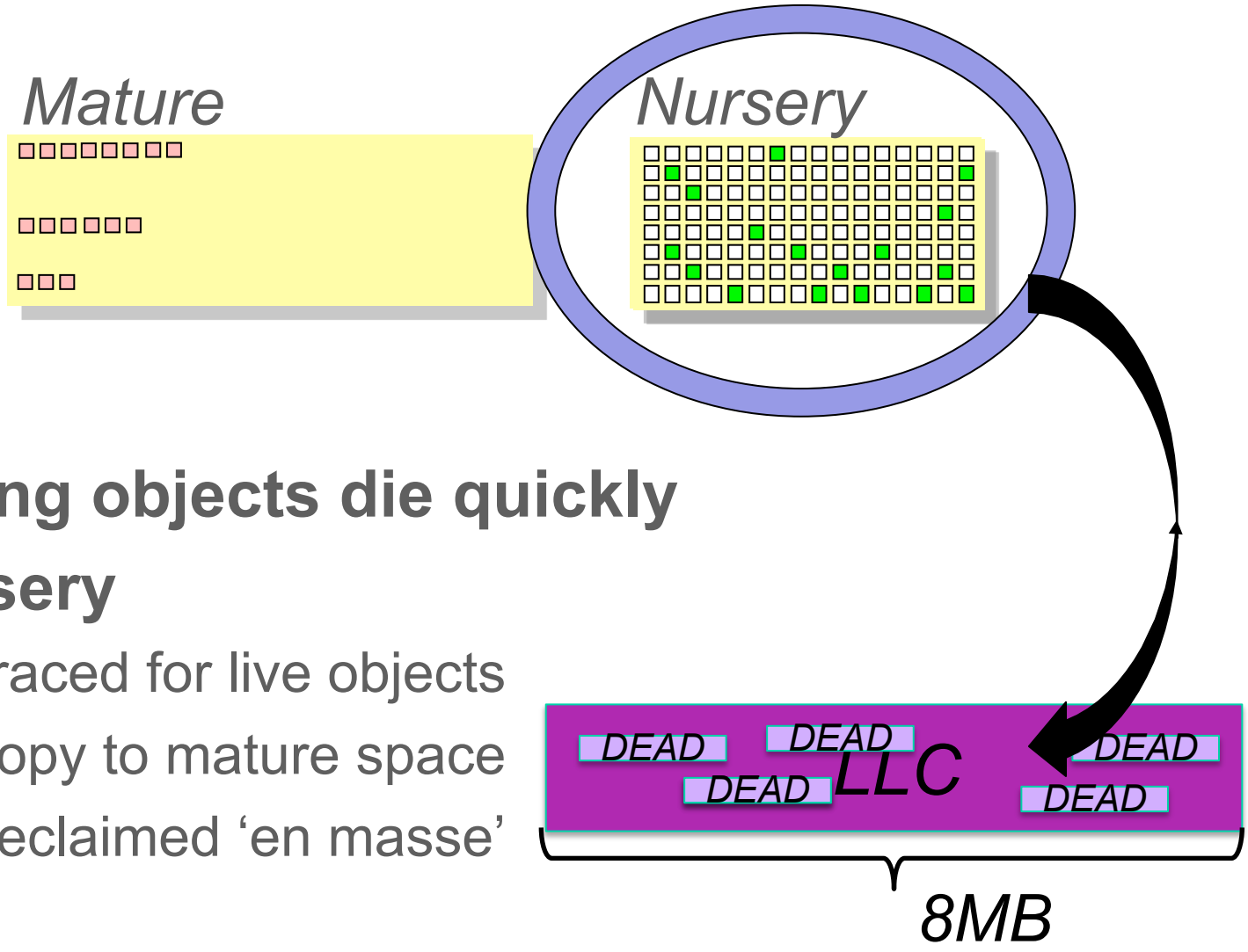
# Problem: Bandwidth & Power Wall



# Cooperative Cache Scrubbing

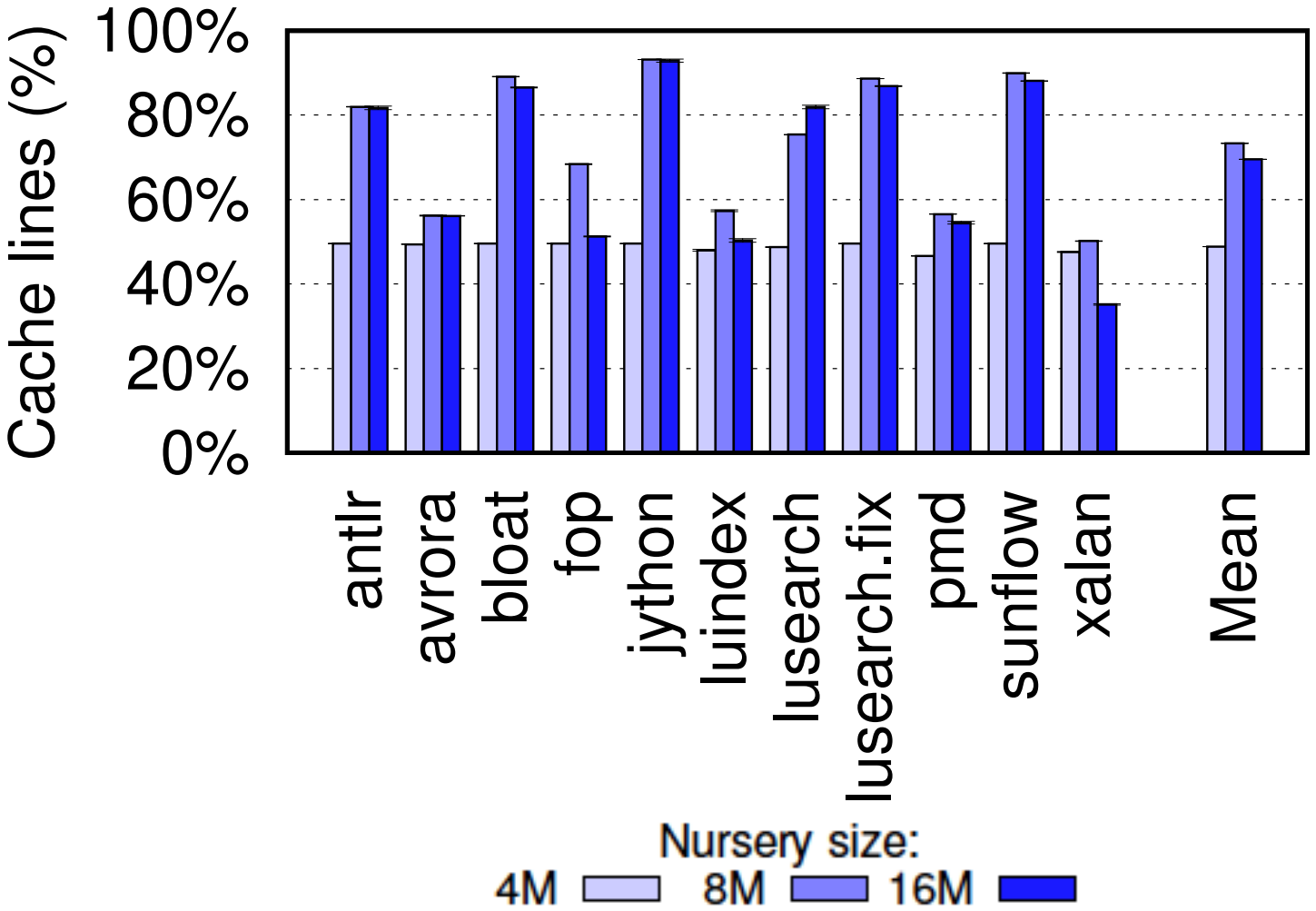


# Generational Garbage Collection

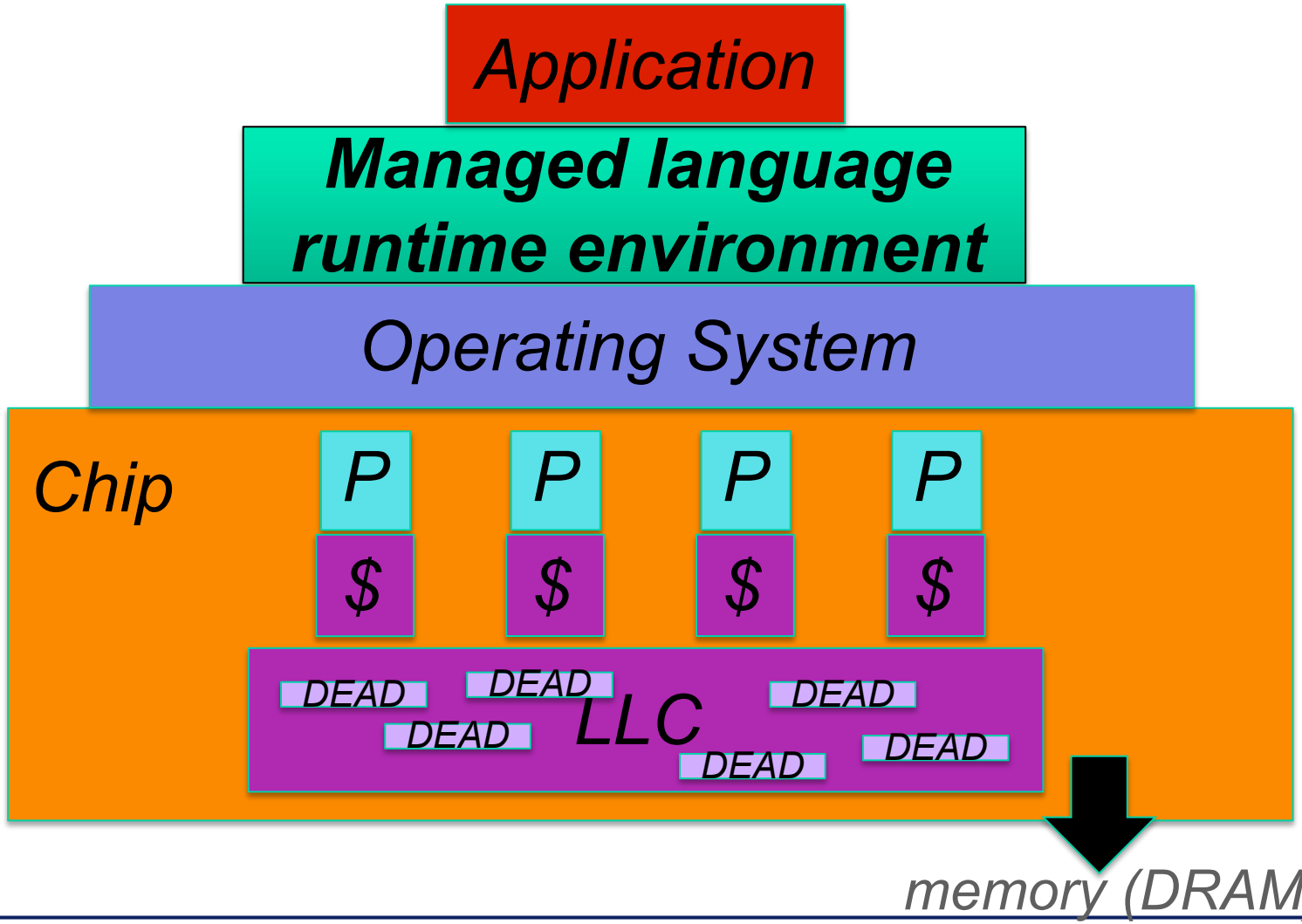



- **Young objects die quickly**
- **Nursery**
  - Traced for live objects
  - Copy to mature space
  - Reclaimed ‘en masse’

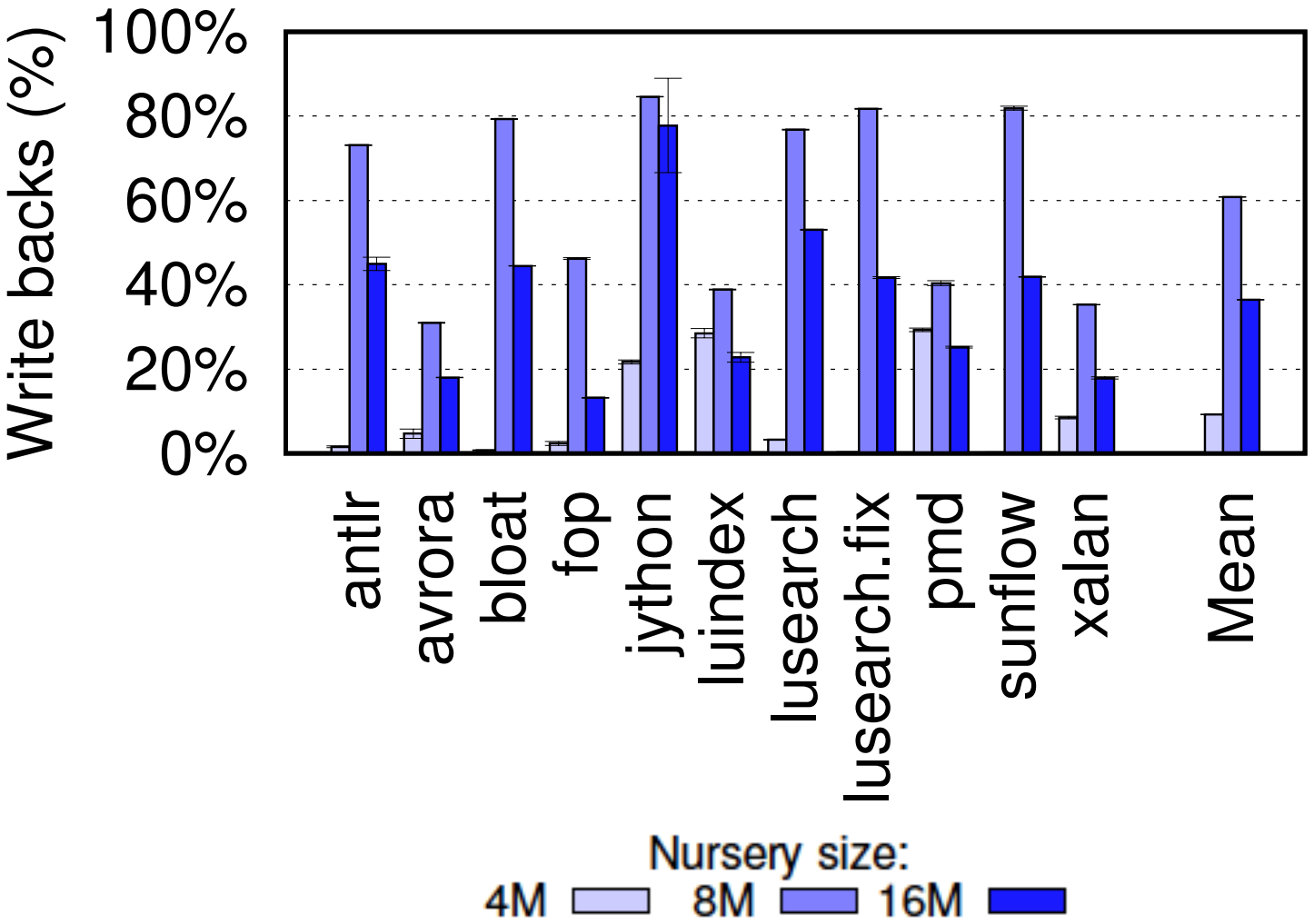
# Dead Lines in LLC (8MB)



# Dead Data Written Back?



# Useless Write Backs (8MB LLC)



# Cooperative Cache Scrubbing

- **Communicate managed language's semantic information to hardware**
- **Caches**
  - 'Scrub' dead lines *writes*
  - Zero lines without fetch *reads*
- **Result**
  - Better cache management
  - Avoid traffic to DRAM
  - Save DRAM energy

# SW-HW Cooperative Scrubbing

- **Software**
  - Identify cache line-aligned dead/zero region
  - Generational Immix collector (stop-the-world)
    - After nursery collection, call scrub instruction on each line in entire range
    - Call zero instructions to zero region (32KB)
- **Hardware**

# SW-HW Cooperative Scrubbing

- **Software**
- **Hardware**
  - Scrubbing (LLC)
    - **clinvalidate:** invalidates cache line
    - **clundirty:** clears dirty bit
    - **clclean:** clears dirty bit, moves line to LRU
  - Zeroing (L2)
    - **clzero:** zero cache line without fetch
  - Modifications to MESI cache coherence protocol
    - Back-propagation from LLC to L1/L2 cache levels
    - Local coherence transitions (no off-chip)

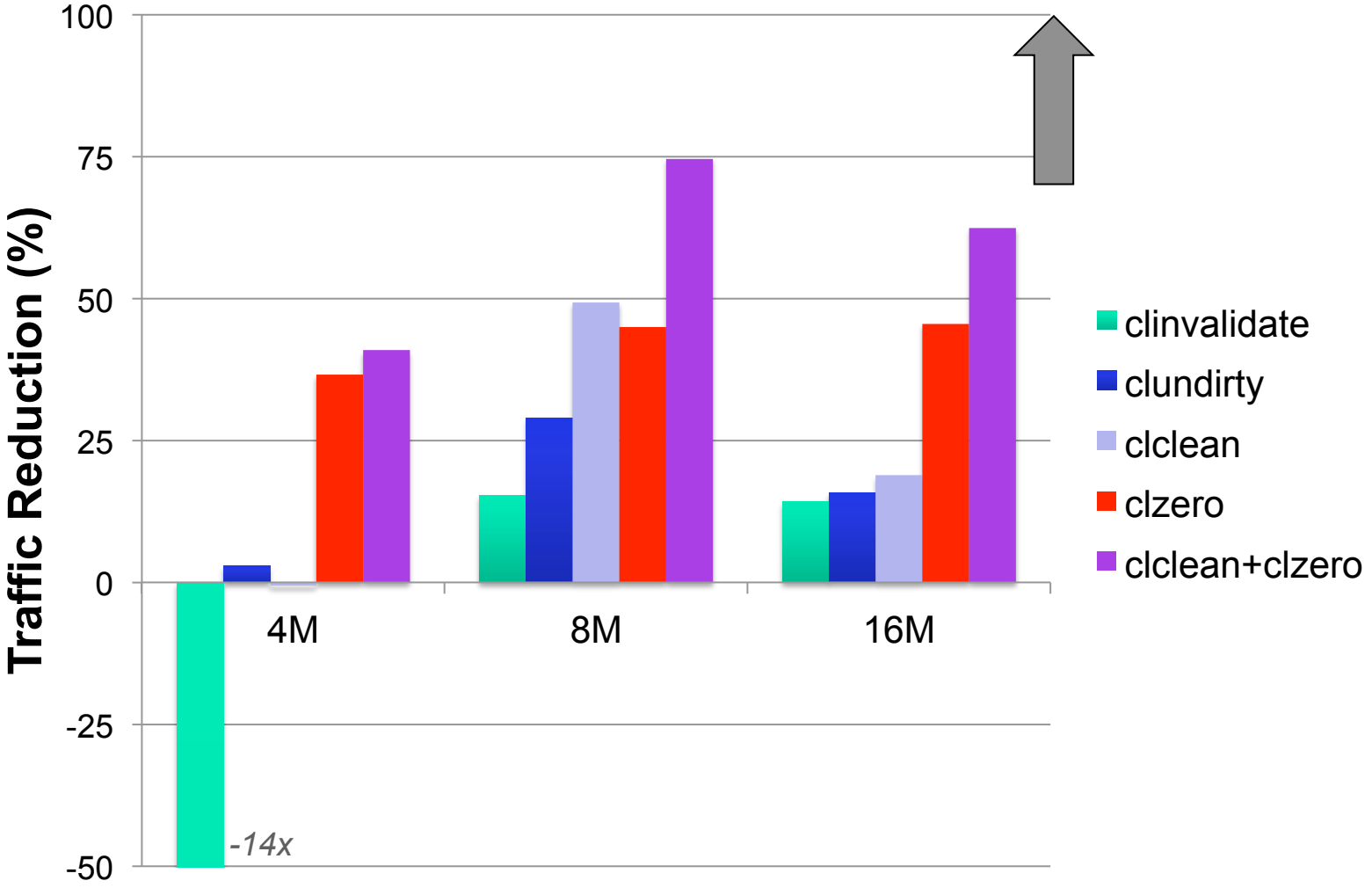
*PowerPC's dcbi, ARM*

*PowerPC's dcbz*

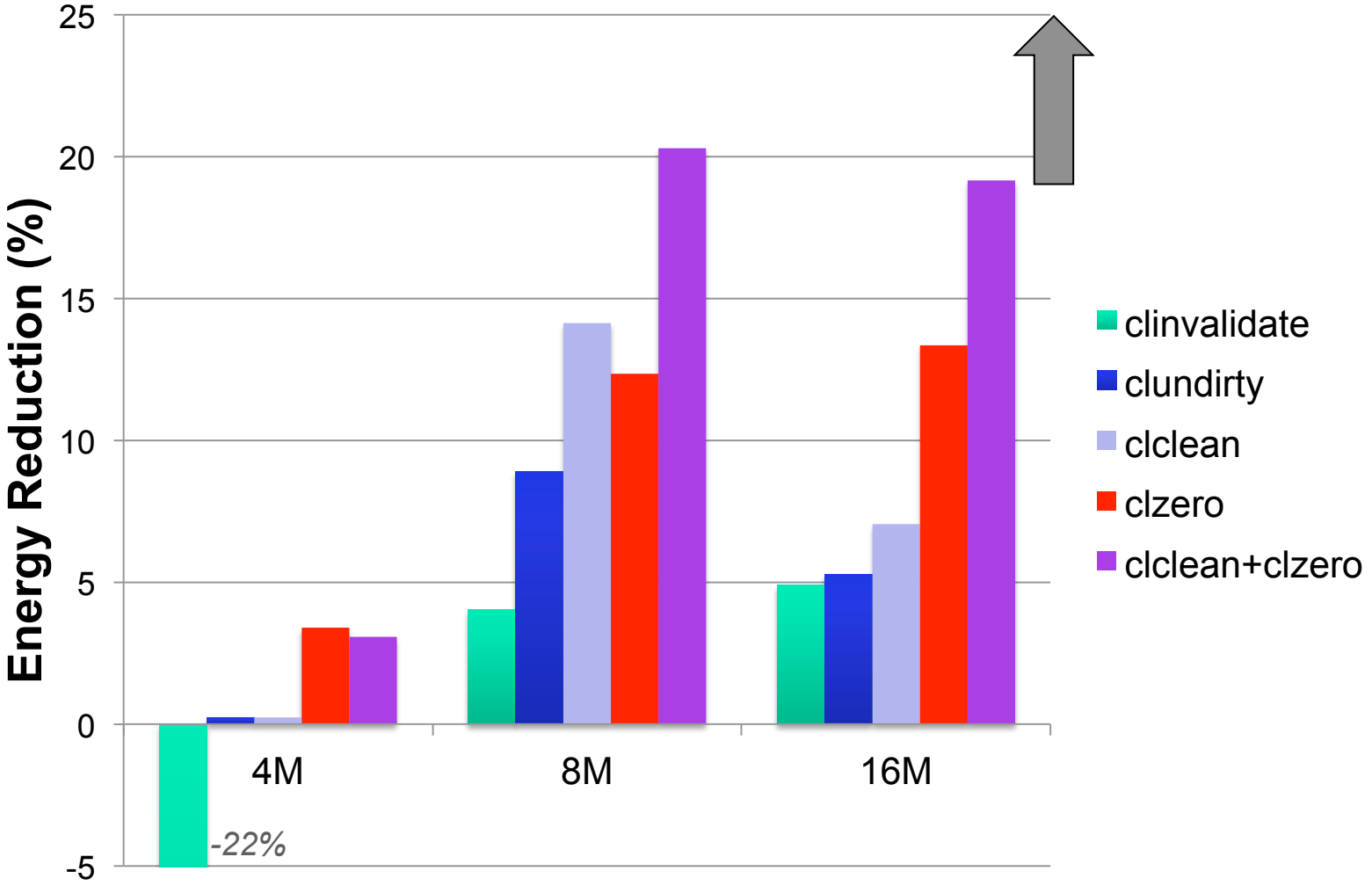
# Methodology

- **Sniper simulator**
  - 4 cores, 8MB shared L3 (LLC), McPAT
- **Jikes RVM 3.1.2 and DaCapo benchmarks**
  - Generational Immix garbage collector
  - 4 application, 4 GC threads

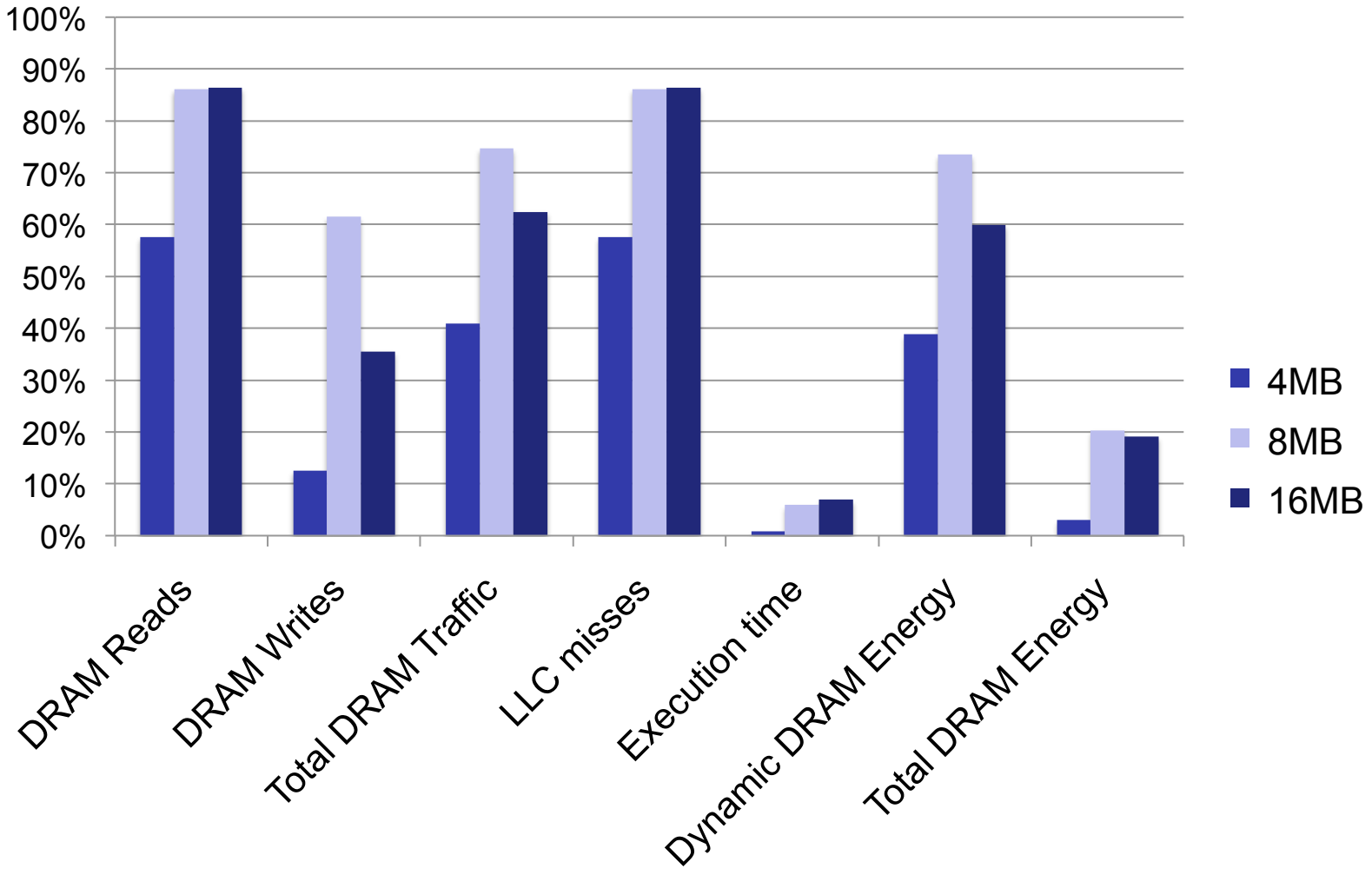
# Total DRAM Traffic



# Total DRAM Energy



# clclean+clzero Improvements

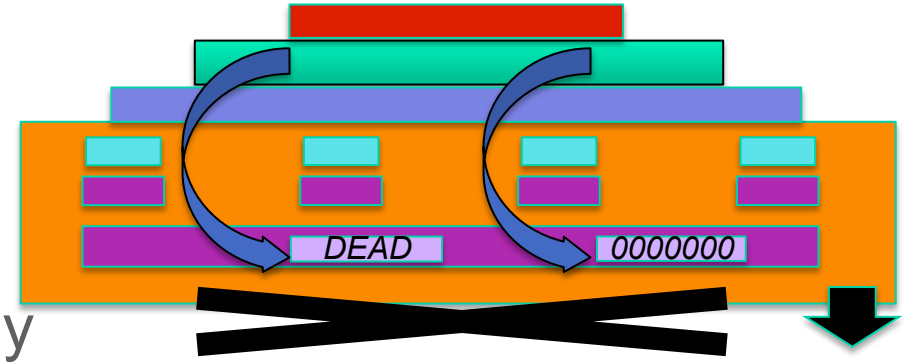


# Related Work

- **Cooperative cache management**
  - ESKIMO by Isen & John, Micro 09
    - Useless reads and writes to DRAM by sequential C programs
    - Reduce energy
    - Require large map in hardware, extra cache bits
  - Wang et al., PACT 02/ ISCA 03; Sartor et al., 05
    - C & Fortran static analysis to give cache hints to evict or keep data
- **Zero initialization [Yang et al., OOPSLA 11]**
  - Studied costs in time, cache and traffic
  - Use non-temporal writes to DRAM, increase bandwidth

# Conclusions

- **Software-hardware cooperative cache scrubbing**
  - Leverages region allocation semantics
  - Changes to MESI coherence protocol
  - New multicore architectural simulation methodology
  - Reductions
    - 59% traffic
    - 14% DRAM energy
    - 4.6% execution time




http://users.elis.ugent.be/~jsartor/